Computers and Electrical Engineering 81 (2020) 106503

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

A DDoS attacks traceback scheme for SDN-based smart city[☆]

Wen Chen, Suchao Xiao, Leijie Liu[*], Xueqin Jiang, Zhangbin Tang

School of Information Science and Technology, Donghua University, Shanghai, 201620, China

ARTICLE INFO

Article history:
Received 12 November 2018
Revised 14 October 2019
Accepted 30 October 2019
Available online 13 November 2019

Keywords:
Smart city
Anomaly detection
Software-defined networking
Distributed denial of service
Traceback

ABSTRACT

Smart City (SC) has brought tremendous opportunities and exciting challenges. Network security is one of the most important challenges in SC networks. Software-Defined Networking (SDN) is more likely to be the target of Distributed Denial of Service (DDoS) attacks due to the risk of a single point of failure. To better defend against DDoS attacks, we focus on tracing the source and propose a statistics-based traceback scheme using the advantages of the SDN architecture. We analyze the changes of flow through the Base Station (BS) nodes to calculate the eigenvalue and establish the anomaly tree. Then prune it with DDoS detection algorithm to get the attack path. Through the experiments, we obtain the optimal parameters to make the proposed scheme more flexible and effective. It shows that the scheme consumes fewer network resources and saves time than traditional ones while keeping the DDoS defense accuracy at a high level.

1. Introduction

The Smart City (SC) approach involves the integration and correlation of information from different fields (e.g. power grid management, water supply, etc.) to optimize asset management. SC is like a popular network of sensors and actuators, where nodes exchange information and take advantage of the power of the Internet of Things (IoT). The IoT has particular applications in public safety as well as other domains such as smart cities, health monitoring, smart homes and environments, smart industry, and various types of pervasive systems [1]. Safety-related Intelligent Transportation Systems (ITSs) are heavily dependent on the Vehicular Area Network (VANET) technology deployed in urban areas, where a massive number of vehicles can be involved. And thus, applying a traffic control system is a necessity in order to collect, process, and control traffic-related activities/actions. Such a system needs sensor networks to help in planning such activities that make the system works better [2]. The IoT will generate a large amount of data, which can be used for security, efficiency and information and entertainment applications and services of urban residents. Managing these huge data through its lifecycle is the foundation of realizing a smart city. In this case, a smart city needs to be supported and integrated by smart communication infrastructure, which defines the role of Software-Defined Networking (SDN) [3].

SDN divides the traditional network into three separate planes (management, control, and data plan) and gives network control to the centralized controller. Compared with traditional network architecture, SDN achieves flexible control of network traffic and makes the network a flexible resource for deployment.

In the smart communication infrastructure for SDN-based SC, as shown in Fig. 1, all IoT devices (such as mobile phones, cars, watches, etc.) continuously produce a large amount of data as terminals. Then all Base Stations (BS) constitute a com-

[☆] This paper is for CAEE special section SI-spsc. Reviews processed and recommended for publication to the Editor-in-Chief by Guest Editor Dr. Weizhi Meng.
[*] Corresponding author.
E-mail address: 2161243@mail.dhu.edu.cn (L. Liu).

https://doi.org/10.1016/j.compeleceng.2019.106503
0045-7906/© 2019 Elsevier Ltd. All rights reserved.

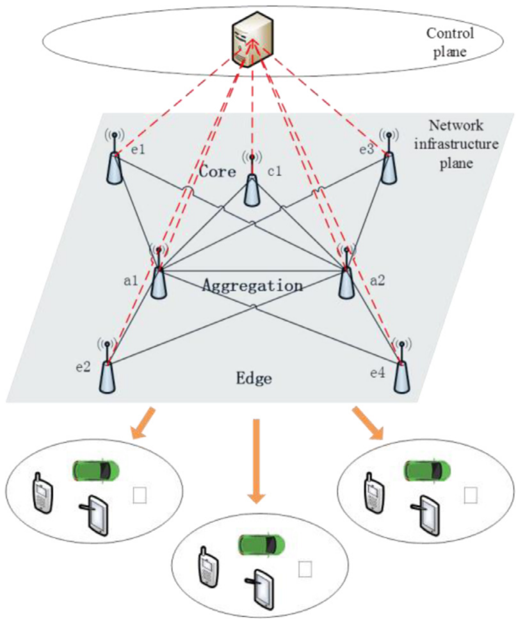

Fig. 1. Network infrastructure for SDN-based SC.

munication network, as the data forwarding plane of SDN network, which is responsible for forwarding the data of the terminal. And the controller located in the control plane monitors and controls each BS finds the BS whose data is abnormal and maintains the reliability and security of the whole communication network.

This smart communication infrastructure can monitor the network situation in the real-time, process a large amount of data sent by the IoT devices effectively. And it is easier to achieve load balance. However, the prominent features of SDN's forwarding and control element separation and centralized control may cause SDN to the risk of a single point of failure. It is more likely to be the target of a Distributed Denial of Service (DDoS) attacks.

Due to a large number of IoT devices and poor security, the devices are more likely to be hijacked, resulting in DDoS attacks. For example, Mirai botnet [4] scans the random public IP address to get the address of the IoT device and tries to infect it. Then the hacker instructs all bots to inject a great number of meaningless traffic into the target server, forcing the target server to fail to respond to the normal network connection, and then in a state of network paralysis. There are many botnets like Mirai such as Hajime botnet, BrickerBot and so on.

In the field of defense against DDoS attacks, the normal steps are detection, traceback, and filtering [5]. However, most researches nowadays generally descript traceback schemes in attack detection algorithms and few are specifically designed for traceback. Considering that traceback is a bridge connecting detection and filtering, in order to decouple them, with the help of the SDN controller's characteristic, this paper proposes a general and lightweight traceback scheme for SC. And our contributions are summarized as follows:

1) As a framework for traceback, we decouple detection, traceback, and mitigation, regard traceback as the main process of DDoS detection and defense, and invoke detection and mitigation algorithms by traceback.
   Therefore, a variety of DDoS detection algorithms can be arbitrarily chosen, if the algorithm is able to distinguish between a flash crowd [6] and DDoS attacks. Meanwhile, some QoS strategies, such as limiting, dropping and so on, could be implemented after finding the attack path.
2) In addition, the existing detection scheme will detect each flow, which results in resource and time consumption. In view of this, our traceback scheme calls detection algorithm only when network traffic is abnormal. By reducing the number of detection, it takes up lower memory and fewer CPU resources to a certain extent.
3) The experiments are performed on the simulation network topology. With the improvement of parameters, the results show that our scheme has the ability to distinguish normal flow from the abnormal flow. And combined with a DDoS detection algorithm, it could discriminate between the flash crowd and DDoS attacks, and make accurate decisions.

The rest of this paper is organized as follows: Section 2 introduces the related works. Section 3 presents the traceback approach design and proposes the anomaly tree's establishment and pruning algorithms. Section 4 shows the simulation network topology, results, and evaluation. Finally, we conclude our work in Section 5.

Table 1
The comparisons of the related paper's advantages and limitation.

Description of the solution	Advantage	Limitation
Proposed an IP traceback mechanism for a large scale distributed online system. [7]	The source of the spoofed packets can also be accurately identified. Quickly raise an alert against potential threats	Time-consuming
A two-step traceback scheme to track the DDoS attack source by dividing the tracing process into two steps. [8]	Low bandwidth consumption, quick convergence speed, the light computational overhead of address recombination	It is hard to obtain the router states of the entire network
Proposed a software-defined security networking mechanism (SDSNM) [11]	Locate quickly and accurately with loose policies	Generality and communication efficiency can be improved
It presents a collaboration of distributed network traffic Monitors and attack Correlations supported by OVS for DDoS detection and containment. [12]	Verification to reduce false alarms Quickly raise an alert against potential threats	Test in small to the medium corporate network
Applied SVM to DDoS traffic identification and employed it on the simulated SDN environment for campus networks as a DDoS detection module. [13]	Improve the effectiveness of the DDoS attack identification method	Once the type of a new flow could not be judged, it had to retrain the model and was rather time-consuming.
Use a third-party application and check the traffic for a specific amount of time to detect DDoS attacks. [14]	More effective It reduces false alarms	Required extra CPU resources. If the sampling rate was set too small, it was easy to make the CPU break down

2. Related work

In Smart City, there are numerous different connected devices and heterogeneous technologies, which causes the security of SC vulnerable when the network is not controlled by attacks. One of the major intimidations to the smart home and urban environments networks is the Distributed Denial of Service attack.

In [7], the authors proposed a traceback mechanism for DDoS attacks using the Extended-DPM scheme. It resolves the disadvantages of existing methods by increasing the throughput of the processing server. A two-step traceback scheme to track the DDoS attack source by dividing the tracing process into two steps was proposed in [8]. Compared with previous algorithms, the two-step traceback scheme has the benefits of low bandwidth consumption, quick convergence speed, the light computational overhead of address recombination, it can decrease the number of packets the path reconstruction needs, and improve the efficiency of path reconstruction, hence making it possible to trace the DDoS attack source rapidly. As for DDoS attack in SC with the traditional network, Vasseur et al. Thereinto, the normal traditional traceback schemes are OPM (Opportunistic Piggyback Marking), AOPM (the Advanced of OPM) and SWAP (the traceback message delivery scheme in Probabilistic Pipelined Packet Marking) [9]. However, it is hard to obtain the router states of the entire network in the traditional network. In addition, Hodo et al. a context-sensitive seamless identity provisioning (CSIP) framework which is a secure mutual authentication approach was proposed in [10]. It can achieve the major security goals of the WMSN in a short time period. It considers several security threats including privileged-insider, replay, user masquerading, secret gateway guessing, and MITM attacks. However, for the Industrial Internet of Things, the DDoS attacks should also be taken into consideration.

In the SDN network, it gives a new way to rethink the defense of DDoS attacks and provides the greater convenience for traceback due to efficiently global network information collection by the SDN controller. The authors of [11] modeled DDoS attacks from the perspective of network architecture. Then a software-defined security networking mechanism (SDSNM) was proposed to remove or restrict these conditions. In [12], the authors use the SDN-supported collaborative approach (SSC) to detect TCP SYN flood attacks on the Global Environment for Network Innovations (GENI). This approach is able to not only quickly raise an alert against potential threats, but also follow it up with careful verification to reduce false alarms. An SDN framework to identify and defend DDoS attacks based on machine learning was proposed in [13]. They applied SVM to DDoS traffic identification and employed it on the simulated SDN environment for the campus network as a DDoS detection module. Roshni Mary Thomas et al. [14] use a traffic monitoring method iftop in the server as a third-party application and check the traffic for a specific amount of time to detect DDoS attack. In [15] the authors propose a cache replacement approach for Fog applications in Software Defined Networks (SDNs). This replacement strategy provides significant availability for the most valuable and difficult to sense data in the SDNs.

As we have seen, the above representative work has some drawbacks, such as additional resource usage, regardless of multiple flow tables on the switch, and is time-consuming. The detailed comparisons of the above paper results can be seen in Table 1. Our job is about how we deal with shortcomings. Further details are as follows.

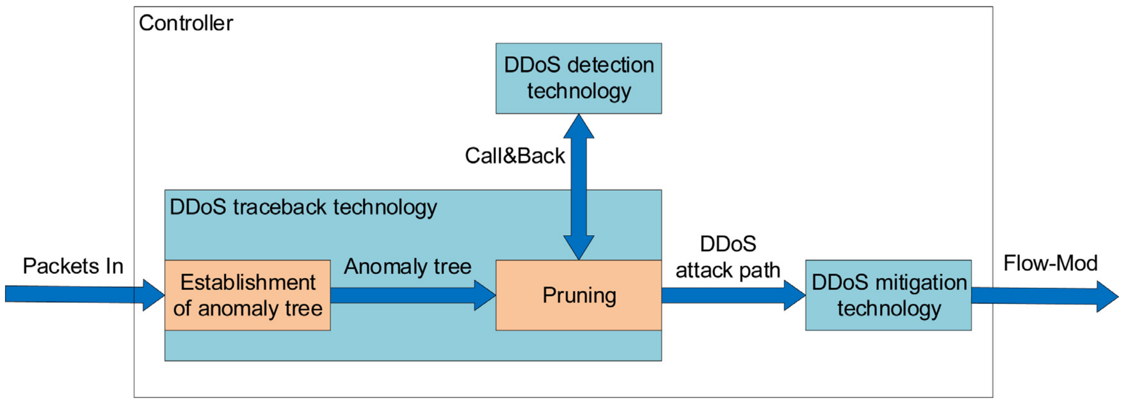
Controller
DDoS detection technology
Call&Back
DDoS traceback technology
Packets In
Establishment of anomaly tree
Anomaly tree
Pruning
DDoS attack path
DDoS mitigation technology
Flow-Mod

Fig. 2. The structure of our scheme.

3. Design of traceback scheme

In this section, we present a statistical-based traceback scheme that can effectively take advantage of the SDN architecture. Fig. 2 shows the workflow of DDoS traceback and the relationship with detection and mitigation technology. According to Fig. 2, the traceback scheme mainly includes two algorithms: anomaly tree's establishment and pruning.

The anomaly tree algorithm calculates the eigenvalues according to the number of packets collected at a fixed time interval and establishes an anomaly tree according to the obtained eigenvalues. it consists of two sub-algorithms: calculate current eigenvalue and build an anomaly tree.

The pruning algorithm traverses the anomaly tree and calls the DDoS detection algorithm to detect each node of anomaly tree, retains the attacked node, and finally gets the attack path.

3.1. Establishment of anomaly tree

Comparing with the traditional network, the SDN controller could get the global network topology easily. It sends Link Layer Discovery Protocol (LLDP) [16] packets to all connected BS nodes for link discovery based on OFPT_PACKET_OUT message. Then the controller identifies and manages the network topology based on the information collected by the LLDP. Taking advantage of this feature, we could monitor the changes in traffic for all the BS nodes in the network topology.

SC is composed of a series of IoT nodes. We assume that under normal circumstances, these nodes are evenly distributed. And each node periodically sends data to the nearest BS. So, without any attacks and flash crowd, the changes of flow through the BS are relatively smooth in a specific sampling frequency. When DDoS attacks or flash crowd appear, the traffic will occur a sudden change in a short period of time. The unexpected change offers us to learn whether the traffic is an anomaly or not. Eigenvalues should be able to correctly represent changes in network traffic, such as throughput [17], entropy [18] and so on. In this paper, we choose throughput as eigenvalues. We define two eigenvalue vectors, which are the eigenvalue vector C composed of m eigenvalues from different BS at the current time, and the eigenvalue vector S composed of non-abnormal eigenvalues recorded at different moments.

We assume that in the current network has m BS nodes. The current network eigenvalues matrix can be defined as $\mathbf{C}=(c_1,c_2,\ldots c_m)^T$. Where c_j represents the eigenvalues of BS j at the current sample time. The last K sample time' normal eigenvalues in BS j can be defined as $\mathbf{S}_j=(f_1,f_2,\ldots,f_K)$.$f_i$ represents the i th normal eigenvalue when network traffic is normal, where $i\in[1,K]$ and K means the number of normal eigenvalues that need to be recorded. Note that $f_K$means the result of the last sample time, f_{K-1} implies the value counted before last. When c_j is judged to be a normal eigenvalue, it will replace f_K at the current sample time, and the original f_K will replace f_{K-1}, and so on, eventually get a new $\mathbf{S}_j$. Naturally, collecting normal eigenvalues from all BS nodes, we can get a $m\times K$ matrix $\mathbf{F}=(\mathbf{S}_1,\mathbf{S}_2,\ldots\mathbf{S}_m)^T$, which represents all normal eigenvalues.

To improve the adaptability and reduce the error probability, we dynamically adjust the average and the threshold of normal eigenvalues by weight. The weight vector is defined as $\boldsymbol{\alpha}=(\alpha_1,\alpha_2,\ldots\alpha_K)^T$, where

$$\alpha_i=\frac{i}{\sum_{i=1}^{K}i},\quad \text{and}\ \sum_{i=1}^{K}\alpha_i=1. \tag{1}$$

Eq. (1) means that the closer to the last sample time, the greater the weight of normal eigenvalues at that sample time. Therefore, the average vector $\mathbf{E}$ can be calculated as

$$\mathbf{E}=(e_1,e_2,\ldots e_m)^T=\mathbf{F}\cdot\boldsymbol{\alpha}, \tag{2}$$

where e_j is the average values of all the normal eigenvalues in BS *j. At the same time, we define the threshold vector* $\boldsymbol{\sigma}$ *as*

$$\boldsymbol{\sigma} = (\sigma_1, \sigma_2, \dots \sigma_m)^T = \frac{|\mathbf{C} - \mathbf{E}|}{\sqrt{K-1}}, \tag{3}$$

where σ_j means the thresholds of all the normal eigenvalues in BS *j*.

At last, we could determine whether the BS j is abnormal as follows:

$$\lceil \max(\mathbf{S}_j) + \sigma_j \rceil < \lfloor c_j \rfloor, \tag{4}$$

where c_j is the currently calculated eigenvalue in the BS *j*. If Eq. (4) is true, it indicates that the *j-th* BS has a traffic mutation. Otherwise, the current traffic is normal and c_j should be added to replace f_K in $\mathbf{S}_j$.

Based on the above, we can design the corresponding algorithm to trace the source of abnormal traffic in the network. The DDoS attack traceback algorithm consists of two sub-algorithms: calculate current eigenvalue and build an anomaly tree.

3.1.1. Calculate current eigenvalue

According to the pre-set sampling time, the SDN controller sends LLDP messages to all the connected BS nodes to confirm whether the number of connected BS nodes changes. If not, the OFPT_PORT_STATUS request message will be sent. After receiving the corresponding response message, the controller will determine whether the current BS has been recorded. And if so, it will obtain the number of bytes of all data flowing through the BS, which is the number of bytes of all packets that have been recorded since the BS was started. Therefore, to facilitate the calculation, the value needs to be processed as:

$$\frac{current\_value - pre\_value}{1024 * 1024 * sampling\_time}. \tag{5}$$

Eq. (5) calculates the flow rate through the BS during the sampling interval and converts it to Mb/s. After calculating the flow rate of all the BS nodes, the current flow rate of all the BS nodes is obtained and assigned to the next calculation. The specific algorithm can be seen in Algorithm 1.

Algorithm 1 Current_Eigenvalue_Calculate.

```
Inputs: Number of BS nodes, Recorded BS nodes' ID.
Outputs: Current eigenvalues matrix C.
1:  m ← number of BS nodes
2:  s ← recorded BS nodes' ID
3:  preValue, C ← a $m \times 1$ matrix with 0
4:  while:
5:    //judge whether the number of BS nodes is changed
6:    controller sends LLDP messages
7:    temp ← get number of connected BS nodes from reply message
8:    if m≠temp then
9:      m ← temp
10:     re-initialize preValue and C
11:   end if
12:   //get current bytes
13:   controller sends OFPT_PORT_STATUS request messages to connected BS nodes
14:   messages ← reply from OFPT_PORT_STATUS messages
15:   for msg in messages do:
16:     for $i=1$:m do:
17:       if msg's ID in s then
18:         bytes ← get bytes from msg
19:         getValue ← (bytes - preValue[$i$])/1024/1024/sampling time
20:         C[i] ← getValue
21:       end if
22:     end for
23:   end for
24:   preValue ← C
25:   Anomaly_Tree ← call Anoamly_Tree_Build to further process
26:   Attack_Tree ← call pruning to further process
27:   wait sampling time
```

3.1.2. Build anomaly tree

Algorithm 2 shows the process of building the anomaly tree, which is called when the Algorithm 1 calculates the new eigenvalues. It should be noted that the BS j added to the anomaly tree refers to the BS nodes' ID. The algorithm receives

Algorithm 2 Anoamly_Tree_Build.

```
Inputs: Current eigenvalues matrix C, Number of BS nodes, Number of recorded value.
Outputs: Anomaly_Tree.
1:  C← Current eigenvalue
2:  m ← number of BS nodes
3:  $K$ ← number of recorded value
4:  F←a $m \times K$ matrix with 0
5:  //initialization threshold
6:  $\boldsymbol{\alpha}$← a $K \times 1$ weight vector
7:  sum ← $K^*(1+K)/2$
8:  for $i = 1{:}K$ then
9:  $\boldsymbol{\alpha}$[i] ← i/sum
10:  end for
11:  //main process
12:  E ←F • $\boldsymbol{\alpha}$
13:  for $j = 1{:}m$ then
14:  if $\lceil \max(\mathbf{S}_j) + \sigma_j \rceil < \lfloor c_j \rfloor$ is true then
15:  add BS $j$ to Anomaly_Tree
16:  else
17:  $\mathbf{S}_j$ ←update $\mathbf{S}_j$ from $c_j$
18:  F[j] ←$\mathbf{S}_j$
19:  end if
20:  end for
```

three parameters: the current eigenvalues, the number of BS nodes in the network, and the number K. Then the algorithm initializes a series of parameters and calculates the weights. After calculating the average and threshold values of the current eigenvalue matrix, the algorithm will call Eq. (4) to determine if the current threshold is exceeded. If it is exceeded, it means the BS has a sudden increase in flow rate and should be added to the anomaly tree. Otherwise, the current network is normal and the eigenvalue matrix needs to be updated for the next calculation. When the algorithm ends, it will output the anomaly tree.

3.2. Pruning

A variety of events occur every day in the city, causing a sudden increase in BS traffic. For example, traffic accidents cause vehicles to be congested, and a large amount of data is sent to the nearest BS, resulting in a sudden increase in BS traffic. The anomaly tree being created in Algorithm 2 may contain these BS nodes by mistaken. To solve this problem, our work needs to detect each node in the anomaly tree and prune extra nodes.

Algorithm 3 Pruning.

```
Inputs: Anomaly_Tree, Current eigenvalues matrix C, Normal eigenvalues matrix F.
Outputs: Attack_Tree.
1:  nodes ← post-order traversal the Anomaly_Tree
2:  //main process
3:  for node in nodes then
4:     call DDoS detection algorithm and return detection_result
5:     if detection_result is DDoS_attack_flow then
6:        Attack_Tree ← add node to Attack_Tree
7:        continue
8:     else
9:        //flash crowd flow
10:         update F with C
11:         take some QoS strategies
12:         send FLOW_MOD message to this BS node
13:     end if
14:  end for
```

In Algorithm 3, we get the nodes from the anomaly tree by post-order traversal and use the DDoS detection algorithm to detect them. Note that, it only requires that the DDoS detection algorithm should discriminate between the flash crowd and DDoS attacks and make accurate decisions. According to the result of the detection algorithm, if it is a flash crowd flow, this node will remove from an anomaly tree. Meanwhile, the current eigenvalue will be updated to the **F**. And some QoS strategies could be implemented, such as load balancing strategies. For the situation of DDoS attacks, this node will be saved. Finally, after the algorithm ends, it will return the DDoS attack path. Subsequently, the DDoS attack mitigation technology will take specific measures against the nodes in the attack path, such as dropping the corresponding flows.

In the implementation process of the whole scheme, the time complexity of the proposed method focuses on two aspects. One is to monitor the entire network topology, which requires traversal communication with all m connected BS nodes. And its time complexity is $O(m)$. The other is that after generating the anomaly tree containing n abnormal nodes, it needs to be pruned by post-order traversal. And its time complexity is $O(n)$, where $n \in [0, m]$. Therefore, the overall time complexity of the whole algorithm is $O(m)$.

Through the anomaly tree's establishment and pruning algorithm, our design tackles the problems as mentioned in Section 2. It avoids the problem of multiple flow tables by using throughput or entropy as an eigenvalue, which considers the number and size of the flow. Compared with detection for each flow, our approach generates the anomaly tree by monitoring network traffic and only detects BS nodes recorded in the anomaly tree. In this way, we reduce the time-consuming of the detection algorithm. This is a qualitative improvement in resource and time consumption.

4. Simulation and evaluation

In this section, we discuss the evaluation and results of our work (Anomaly Tree). We evaluate the performance of the proposed approach and additionally compare the results against some schemes (OPM, AOPM, SSC, SVM, and SDSNM) mentioned in Section 2.

4.1. Simulation network topology

In this paper, Mininet [19] is chosen as the implementation of network topology. It is a network emulator that runs a collection of end-hosts, switches, routers and links on a single Linux kernel. It is designed to easily create virtual SDN consisting of an Open-Flow controller, a flat Ethernet network of multiple OpenFlow enabled Ethernet, switches and multiple hosts connected to those switches for highly flexible custom routing and Software-Defined Networking.

Compared to other network emulators, such as NS2 and Opnet. Mininet combines many of the best features of emulators, hardware testbeds, and simulators. It has a number of advantages. Flexibility, that is, new topologies and new features can be set in software, using programming languages and common operating systems. Applicability, correctly implementations conducted in prototypes should also be usable in real networks based on hardware without any changes in source codes. Interactivity, management and running the simulated network must occur in real-time as if it happens in real networks. Scalability, the prototyping environment must be scaling to large networks with hundreds or thousands of switches on only a computer. Realistic, the prototype behavior should represent real-time behavior with a high degree of confidence, so applications and protocols stacks should be usable without any code modification [20].

We select Ryu [21] as the SDN controller. Ryu can easily control thousands of OpenFlow BS nodes in logic while providing users with a flexible programming network control interface. Meanwhile, it develops based on Python and is easy to use as a platform for building SDN applications.

In [22], Dayal and Srivastava analyzed the behavior of DDoS attacks and classified attack types in SDN. They even listed the types of common DDoS attack flows for different attack locations. Refer to their study, we select several types of DDoS attacks (Ping flood, TCP_SYN flood, UDP flood, HTTP flooding, DNS reflection, and NTP amplification).

For topology, as shown in Fig. 3, we evaluate the proposed approach in a tree topology, which consists of *7* BS nodes (core node-*c1*, aggregation nodes-*a1 a2*, edge nodes-*e1 e2 e3 e4*). And each BS node contains thousands of active IoT nodes. These nodes communicate with BS all the time.

4.2. Evaluation and discussion

In our experiment, we randomly select a DDoS attacking user and target from IoT nodes. Meanwhile, we prepare several DDoS attack flows, which are randomly assigned to six types of attack (Ping flood, TCP_SYN flood, UDP flood, HTTP flooding, DNS reflection, and NTP amplification). For NTP amplification, In [23], the authors had demonstrated that at least 14 UDP-based network protocols or service implementations including network services (such as NTP, SNMP, SSDP and NetBios),

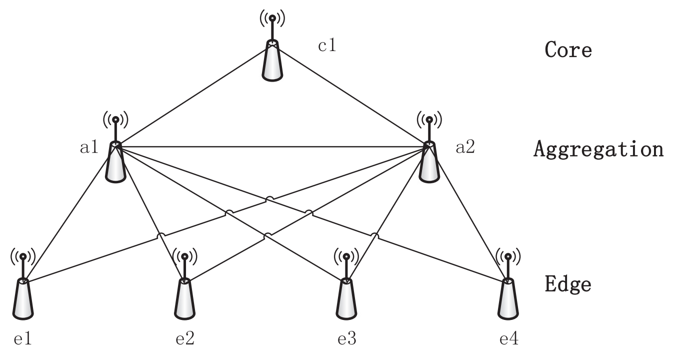

Fig. 3. Simulation network topology.

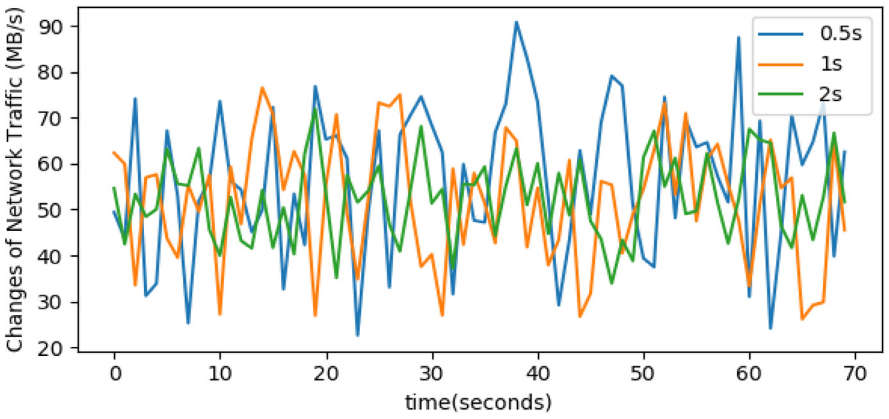

Fig. 4. The change of network traffic at a different sampling frequency.

Table 2
Times of DDoS attacks.

Time (seconds)	Types of flow
0–300	Normal flows
300–350	DDoS attacks
350–880	Normal flows
880–934	Flash crowd
934–1178	Normal flows

Table 3
Index values at different frequency.

Frequency(*s*)	0.5*s*	1*s*	2*s*
Mean(*MB/s*)	56.69	51.91	52.32
Variance	260.23	178.57	76.05
Max(*MB/s*)	90.77	76.51	71.91
Min(*MB/s*)	22.63	26.13	33.92

legacy services (CharGen and QOTD), peer-to-peer (P2P) filesharing networks (BitTorrent and Kad), game servers (Quake 3 and Steam) and P2Pbased botnets are susceptible to amplification abuse. they also proposed methods to detect real-world DRDoS attacks and analyzed more than 130 real-world attacks. For the DNS reflection, the authors in [24] presented a flavor of the DNSSEC-powered amplification attack that takes advantage of the vast number of DNS forwarders out there. The attack relies on DNS forwarders and takes advantage of large DNSSEC RRs.The attacker enjoys anonymity. They assessed the strength of the attack and witnessed a maximum amplification factor of 44.

Table 2 presents the occurrence time and duration of DDoS attacks and flash crowds.

We mainly show the experimental results of the traceback scheme from three aspects: the selection of key parameters, the implementation process of the traceback scheme, and the performance analysis of different traceback algorithms.

4.2.1. Selection of key parameters

It can be shown from Algorithm 1 and 2 that our traceback scheme based on the anomaly tree needs to determine the sampling frequency and the value of the parameter K in advance. The selection of these values has a certain influence on the experimental results. The difference in sampling frequency can result in a sudden increase or a sudden drop in the originally smooth network traffic.

Fig. 4 shows the change in network traffic at different sampling frequencies when the flow rate in the network is 50 MB/*s*. It can be clearly seen from the figure that at a sampling frequency of *0.5 s*, the variation curve fluctuates greatly and the maximum phase difference can reach *70*. When the sampling frequency is *2 s*, the curve fluctuation is relatively flat, and the latter value does not exceed *1.5* times the previous value, which is a normal range. Meanwhile, Table 3 also shows the index values at the different sampling frequencies. The variance is the smallest at the sampling frequency of *2 s*, and the maximum value is within the range of *1.5* times the mean. Therefore, this paper selects the sampling frequency of *2 s* to sample the traffic in the SDN network.

In our model, K is another important parameter. It is directly related to the threshold matrix $\boldsymbol{\alpha}$ and the dimension of the eigenvalue matrix $\mathbf{F}$. Many different eigenvalues will indirectly affect the size of the threshold $\boldsymbol{\sigma}$ through Eq. (3), and ultimately affect the judgment result of Eq. (4) on whether the BS is abnormal. As is shown in Fig. 5, with the value of K

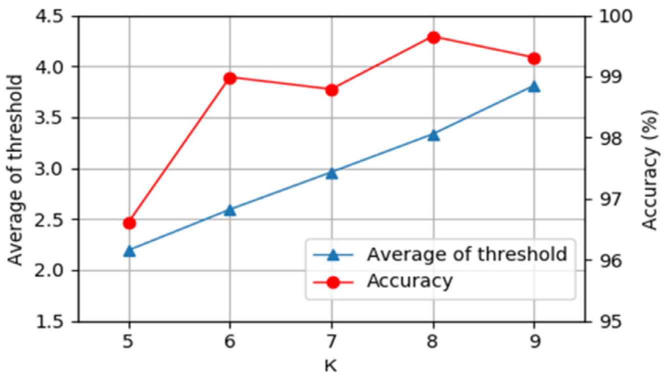

Fig. 5. The average threshold and accuracy with different values of *K*.

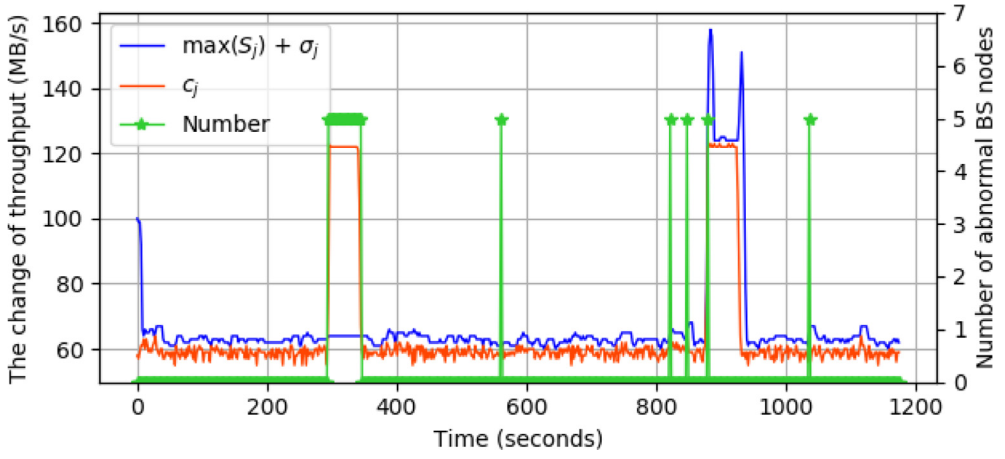

Fig. 6. The change of network load and the number of abnormal BS nodes.

increases, the average change of threshold grows at a near-linear trend. Although the accuracy rate has fluctuations, it is rising overall. Therefore, from a comprehensive perspective, the *K* chosen in this paper is *6*.

4.2.2. Implementation process

We use the Python-based Scapy library to generate the six DDoS attack flows, set the bandwidth of network topology to *1Gbit*, initialize the eigenvalue matrix to half of the bandwidth. Meanwhile, we control the normal flows at 60 MB/*s*. As it is shown in Fig. 6, the traceback scheme can make different processing according to the different types of traffic in the SDN network.

First, the values $\max(\mathbf{S}_j) + \sigma_j$ have a downward impulse to adapt to the current traffic. When DDoS attacks happen at 300 s, the anomaly tree's establishment algorithm records the abnormal BS nodes. Then, the pruning algorithm determines that the current traffic is a DDoS attack and keep the number of abnormal BS nodes to *5*. Subsequently, the DDoS attacks finish at 350 s. So, the network traffic starts to decrease to normal level (about 60 MB/*s*) after that moment, and nodes in the anomaly tree are cleared.

At 880 s, the anomaly tree records the abnormal BS nodes with the sudden increase of network traffic. It activates the pruning algorithm and verifies that the current traffic is just flash crowd flows. Then, it removes the nodes from the anomaly tree and updates the eigenvalue list. Therefore, in Fig. 6, the number of abnormal BS nodes only records one outlier at 880 s. And the values $\max(\mathbf{S}_j) + \sigma_j$ have an upward impulse due to the sudden increase of c_j. At the end of the flash crowd, it quickly falls back to the normal level.

4.2.3. Performance analysis

In the experiment, we use the same network topology as shown in Fig. 3 and set the number of the attacker to 1. We study the performance of the traceback scheme from the following two aspects: time consumption and CPU utilization.

Since our traceback scheme needs to call detection algorithms for pruning, we count the sum of detection and traceback time of several common methods. And the experimental results are shown in Fig. 7. We observe that traditional methods

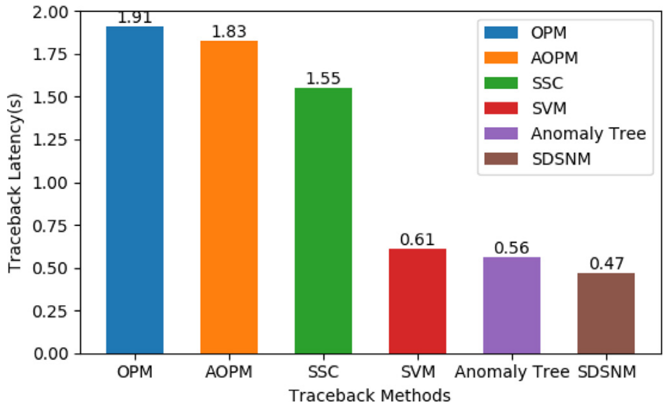

Fig. 7. Time consumption of one traceback for several methods.

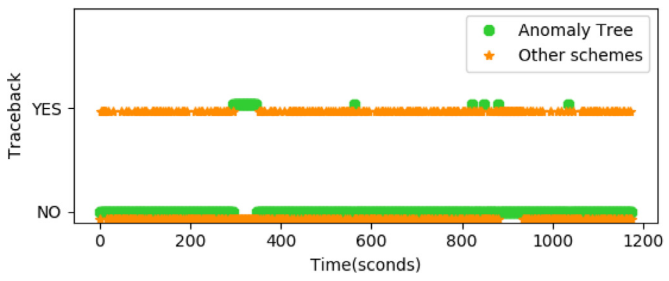

Fig. 8. Whether it is traceback in the experiment.

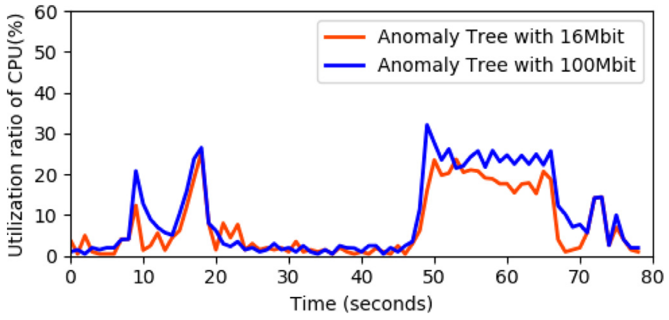

Fig. 9. The utilization ratio of the CPU of the controller over time.

(OPM, AOPM in [7]) take more time since they are not well adapted to the SDN environment. SSC scheme in [15] spends more than *1* s on the Monitor, resulting in a longer overall time. In addition, SVM in [16], SDSNM in [14] and our scheme (Anomaly Tree) make full use of the characteristics of SDN and consume almost the same time.

To further measure the performance between our work and other schemes (SVM, SDSNM), we make more efforts to analyze the traceback times and CPU usages. In Fig. 8, there is a significant difference in traceback times. Our scheme can reduce the traceback times by only detecting the abnormal flows. But other schemes need to detect each new flow.

In terms of CPU usages, we adopt a dual-core CPU (CPU0 and CPU1) and compare the utilization ratio of the CPU of our scheme.

In Fig. 9, we study the impact of different bandwidth in our approach. For *16 Mbit*, the CPU usage is a little higher from 7 s to 10 s. In the period, the maximum utilization ratio of CPU is about *12%*. At 15 s, the CPU usage both rapidly rises to *25%* due to receiving PACK_IN messages. Later, as DDoS attack begins at 48 s and continues to 68 s, the CPU usage only increases to *20%* for traceback in our scheme. After the attack, the CPU usage returns to the normal level. Furthermore, with the increase of bandwidth (from *16Mbit* to *100Mbit*), the average utilization rate of CPU little changes. It can stabilize DDOS attacks with different bandwidth.

In view of this, it is concluded that our scheme can effectively defend DDoS attacks with fewer traceback times and less utilization of CPU resources.

Our solution is based on controllers, but for multiple controllers, we have made some research results and published relevant papers [25]. In our previous work, we proposed an approach named community detection controller deployment (CDCD) to solve the controller placement problem (CPP) in large-scale networks. We regarded the network topology of the controller to be deployed as a network composed of multiple communities and then selected a suitable position in each community to place the controller.

5. Conclusion

In SC, a large number of IoT devices will continuously generate large amounts of data. Managing these huge data through its lifecycle is the foundation of realizing a smart city. SDN's capability of hierarchical control of network security meets the need of building architecture to carry thousands of applications for SC. However, the prominent features of SDN make it more likely to be the target of DDoS. In this paper, we focus on traceback approaches of DDoS attack in SDN-based SC. By analyzing relevant papers in recent years, it is found that the existing methods consume more time and resources, and the overall method is not flexible enough. Therefore, we propose a lightweight traceback scheme based on Anomaly Tree. This scheme establishes Anomaly Tree by analyzing the fluctuation of network traffic, and call various detection algorithms that meet the requirements to prune and finally get the attack path. Through the experiments, we obtain the optimal parameters to make the proposed scheme more flexible and effective.

For future work, we plan to optimize the approach for adaptive with all types of DDoS attacks. Meanwhile, we will consider the cases that some of the BS nodes could provide false data to the controller.

Declaration of Competing Interest

The authors declared that they have no conflicts of interest to this work.

Acknowledgements

This work was supported by the National Nature Science Foundation of China under Grant No. 61501108.

References

[1] Al-Turjman F. QoS–aware data delivery framework for safety-inspired multimedia in integrated vehicular-iot. Comput Commun 2018;121:S0140366417306060.
[2] Al-Turjman F, Alturjman S. Confidential smart-sensing framework in the iot era. J Supercomput 2018;74.
[3] Kreutz D, Ramos FMV, Verissimo PE, Rothenberg CE, Azodolmolky S, Uhlig S. Software-defined networking: a comprehensive survey. Proc IEEE 2015;103:14–76. doi:10.1109/JPROC.2014.2371999.
[4] Kolias C., Kambourakis G. DDoS in the iot: 2017. doi:10.1109/MC.2017.201.
[5] Papalambrou A, Stefanidis K, Gialelis J, Serpanos D. Detection, traceback and filtering of denial of service attacks in networked embedded systems. In: Proc. 9th Work. Embed. Syst. Secur. WESS 2014; 2014.
[6] Jinbo S, Alghazawy BA, Fujita S. Batch-based flash crowd relaxation in cloud-assisted P2P live streaming. In: Proc. - 18th IEEE/ACIS Int. Conf. Softw. Eng. Artif. Intell. Netw. Parallel/Distributed Comput. SNPD 2017; 2017. p. 401–6. doi:10.1109/SNPD.2017.8022753.
[7] Soundar Rajam VK, Mercy Shalinie S. A novel traceback algorithm for ddos attack with marking scheme for online system. In: Int. Conf. Recent Trends Inf. Technol. ICRTIT 2012. IEEE; 2012. p. 407–12. doi:10.1109/ICRTIT.2012.6206751.
[8] Qu Z, Huang C, Liu N. A novel two-step traceback scheme for ddos attacks. In: Proc. - 2008 2nd Int. Symp. Intell. Inf. Technol. Appl. IITA 2008, 1. IEEE; 2008. p. 879–83. doi:10.1109/IITA.2008.102.
[9] Cheng L, Divakaran DM, Lim WY, Thing VLL. Opportunistic piggyback marking for ip traceback. IEEE Trans Inf Forensics Secur 2016;11:273–88. doi:10.1109/TIFS.2015.2491299.
[10] Al-Turjman F, Alturjman S. Context-Sensitive access in industrial internet of things (IIoT) healthcare applications. IEEE Trans Ind Informatics 2018;14:2736–44. doi:10.1109/TII.2018.2808190.
[11] Wang X, Chen M, Xing C. SDSNM: a software-defined security networking mechanism to defend against DDoS attacks. In: 2015 Ninth Int. Conf. Front. Comput. Sci. Technol.. IEEE; 2015. p. 115–21.
[12] Chin T, Mountrouidou X, Li X, Xiong K. An SDN-supported collaborative approach for DDoS flooding detection and containment. Proc - IEEE Mil Commun Conf MILCOM 2015; 2015. Decem:659–64. doi:10.1109/MILCOM.2015.7357519.
[13] Yang L, Zhao H. DDoS attack identification and defense using SDN based on machine learning method. In: Proc. - 2018 15th Int. Symp. Pervasive Syst. Algorithms Networks, I-SPAN 2018. IEEE; 2019. p. 174–8. doi:10.1109/I-SPAN.2018.00036.
[14] Thomas RM, James D. DDOS detection and denial using third party application in sdn. In: 2017 Int. Conf. Energy, Commun. Data Anal. Soft Comput. ICECDS 2017. IEEE; 2018. p. 3892–7. doi:10.1109/ICECDS.2017.8390193.
[15] Al-Turjman F. Fog-based caching in software-defined information-centric networks ☆. Comput Electr Eng 2018;69:54–67.
[16] Nguyen TH, Yoo M. Analysis of link discovery service attacks in SDN controller. Int Conf Inf Netw 2017:259–61. doi:10.1109/ICOIN.2017.7899515.
[17] Huang Y, Sun H, Chao HJ, Chao X. Non-negative increment feature detection of the traffic throughput for early ddos attack. In: Proc - Int Conf Signal Image Technol Internet Based Syst SITIS 2007; 2007. p. 121–6. doi:10.1109/SITIS.2007.122.
[18] Mousavi SM, St-Hilaire M. Early detection of DDoS attacks against SDN controllers. In: 2015 Int Conf Comput Netw Commun ICNC 2015; 2015. p. 77–81. doi:10.1109/ICCNC.2015.7069319.
[19] Pooja, Sood Manu. SDN and mininet: some basic concepts. Int J Adv Netw Appl 2015;7:2690–3.
[20] Keti F, Askar S. Emulation of software defined networks using mininet in different simulation environments. In: Proc. - Int. Conf. Intell. Syst. Model. Simulation, ISMS, 2015; 2015. p. 205–10. Octob. doi:10.1109/ISMS.2015.46.
[21] Kubo R, Fujita T, Agawa Y, Suzuki H. Ryu SDN framework-open-source SDN platform software. NTT Tech Rev 2014;12.
[22] Dayal N, Srivastava S. Analyzing behavior of DDOS attacks to identify DDOS detection features in SDN. In: 2017 9th Int Conf Commun Syst Networks, COMSNETS 2017; 2017. p. 274–81. doi:10.1109/COMSNETS.2017.7945387.
[23] Rossow CBT-N&; DSSS. Amplification hell: revisiting network protocols for ddos abuse, 2014.
[24] Anagnostopoulos M, Kambourakis G, Kopanos P, Louloudakis G, Gritzalis S. DNS amplification attack revisited. Comput Secur 2013;39:475–85.

[25] Chen W, Chen C, Jiang X, Liu L. Multi-Controller placement towards sdn based on louvain heuristic algorithm. IEEE Access 2018;6:49486–97. doi:10.1109/ACCESS.2018.2867931.

Wen Chen, received Ph.D. degree from Compute Science and Engineering Department of Shanghai Jiao Tong University. She is currently an associate professor at College of information science and technology of Donghua University. Her research interests are resource management, optimization techniques, feedback control techniques and their applications in wireless communications.

Suchao Xiao, received the B.S. degree from Nanjing University of Science and Technology in electronic information science and technology. He is currently pursuing the M.S. degree in control science and engineering from Donghua University, Shanghai, China. His-research interests include SDN, 5 G, and IoT.

Leijie Liu, received the B.S. degree from Zhejiang Sci-Tech University in electronic information science and technology. He is currently pursuing the M.S. degree in control science and engineering from Donghua University, Shanghai, China. His-research interests include SDN and IoT. Email: 2161243@mail.dhu.edu.cn

Xueqin Jiang, received the M.S and Ph.D degree from Chonbuk National university, Jeonju, Korea, in electronics engineering. He is now an Associate Professor at School of Information Science and Technology, Donghua University, Shanghai, China. His-main research interests include LDPC codes, physical-layer security and wireless communications.

Zhangbin Tang, received the B.S. degree in communication engineering from Wuhan Institute of Technology. He is currently pursuing the M.S. degree in information and communication engineering from Donghua University, Shanghai. His-research interests include edge cache and its applications in Wireless networks.